In the Specification:

Please amend the paragraph beginning on page 9, line 8, as follows:

FIG. 1 shows one embodiment of a communication system 1000 that may include an error code encoder 100 and an error code decoder 300. Communication system 1000 may include a transmitter 1001 and a receiver 1002. Transmitter 1001 may be configured to encode data into a code word using encoder 100. The code word may be transmitted to receiver 1002 via transmission medium 1003. Receiver 1002 may receive the transmitted code word. Using decoder 300, receiver 1003~~2~~ may extract the original data bits sent by transmitter ~~902~~1001 and detect whether any errors occurred during transmission. If a single-bit error occurred in a data bit, decoder ~~907~~300 may correct the erroneous bit. If the error code being used is a SECDED code or a code that provides even higher levels of error detection/correction, decoder ~~907~~300 may also detect whether any multi-bit errors occurred during transmission of the code word. The transmitter 1001 and receiver 1002 shown in FIG. 1 may each also include a tester to test encoder 100 and decoder 300. The testers may operate as described below with reference to FIGs. 3-10.

Please amend the paragraph beginning on page 13, line 17, as follows:

Check bit generator 102 generates check bits 14 from data bits 12. FIG. 4B shows an example of a parity check matrix 20 that may correspond to the encoding used by one embodiment of check bit generator 100. FIG. 4B also shows the check bit equations 22 corresponding to the parity check matrix 20. As labeled, each column in parity check matrix 20 may correspond to a data bit or a check bit. The check bit equations 22 correspond to each row of the parity check matrix. For example, check bit c0 may be defined to equal ~~a0 XOR a1 XOR a2~~d0 XOR d1 XOR d2 based on the pattern of ones and zeros in the first row of matrix 20 (each data bit whose column has a one in the first row is included in the equation for c0). Since each column is unique and non-zero, each check bit 14 depends on a unique subset of the data bits 12.

Please amend the paragraph beginning on page 15, line 17, as follows:

In one embodiment, a check bit generator may be tested by using a subset of all of the potential data bit combinations as the set of test data. FIG. 4D illustrates a table ~~30A~~40A of one such subset of the data bit combinations. The subset of data bit combinations shown in table ~~30A~~40A may be selected so that each data bit d0-d3 has a value of '0' in at least one combination and a value of '1' in at least one combination. This way, the check bit generator is tested using both possible values ('0' and '1') of each data bit. As shown by table 30, for the illustrated selection of input values, the check bit generator 102 should generate each value ('0' or '1') of each check bit ~~12~~14 at least once, also. Selecting the subset of data bit combinations may thoroughly test the functionality of the check bit generator 102. For example, in some embodiments of check bit generator 102, choosing the subset of data bit combinations so that each data bit assumes each possible value at least once and so that each possible value of each check bit is generated at least once may test logic within check bit generator 102 for stuck-at-0 and stuck-at-1 faults.

Please amend the paragraph beginning on page 16, line 10, as follows:

While other subsets of the data bit combinations may be selected, it may be desirable to select one of the illustrated subsets from FIGs. ~~1D and 1E~~4D and 4E. The subsets shown in FIGs. 1D and 1E may be created by setting a set of initial data bits to all zeros or all ones and then "walking" or shifting a 1 or a 0 across each data bit position d0-d3. After each step or shift, a different one of the data bit combinations is generated. In some embodiments, this may simplify the creation of the set of test data.

Please amend the paragraph beginning on page 20, line 5, as follows:

FIG. 6A also shows an embodiment of a tester 200 that may be used to test the operation of the decoder 300. Tester ~~200~~300 may be a hardware tester in some embodiments. In other embodiments, all or part of tester 200's functionality may be implemented in software. Tester 300 may include a controller 204 configured to control comparison logic 206 and test code word generator 250. Test code word generator 250 may be configured to generate test code words and to provide those test code words to decoder 300. Exemplary sets of test code words that may be generated are illustrated in and discussed with reference to FIGs. ~~6C and 6D~~FIG 6. Comparison logic 206 may be configured to receive the corrected data bits output from decoder 300 in response to each test code word and to compare those corrected data bits to a set of known correct data bits for that test code word. Comparison logic 206 may also receive the single error corrected/detected signal 18 and compare that signal to a known correct signal for each test code word. Based on the result of the comparison(s), comparison logic 206 may output a verification signal 222 indicating whether the decoder 300's output is correct. FIG. 7 describes the operation of a tester 200 according to one embodiment.